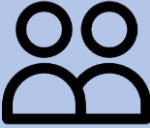
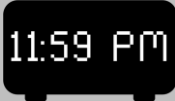

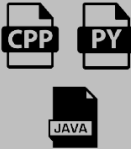




Taller en Sala Nro. 2 Recursión

	Trabajo en Parejas		Hoy, plazo máximo de entrega		Docente entrega código suelto en GitHub
	Sí .cpp, .py o .java		No .zip, .txt, html o .doc		Alumnos entregan código suelto por GitHub

Ejercicios a resolver



Este algoritmo se puede usar para determinar el tamaño de las baldosas para el piso en videojuegos como The Sims 3 de tal forma que se ajusten a las dimensiones del piso sin dejar sobrantes.

1. El algoritmo de Euclides calcula el máximo común divisor de dos enteros (MCD). El MCD de dos números es el entero más grande que divide a ambos. **Como un ejemplo**, el máximo común divisor de 102 y 68 es 34.

El algoritmo de Euclides propone calcular el MCD de esta forma:

“El MCD de p y q es el mismo que el MCD de $p \% q$ y q ”, donde $\%$ es la operación de módulo. Asuman que el usuario siempre entrega p y q de tal forma que $p > q$.

Implementen un método que calcule el algoritmo de Euclides



Una variante de este problema fue propuesto en la maratón de Dropbox del 2011, ve <http://www.skorks.com/2011/02/algorithms-a-dropbox-challenge-and-dynamic-programming/>

2. Pepito escribió un algoritmo que, dado un arreglo de enteros, decide si es posible escoger un subconjunto de esos enteros, de tal forma que la suma de los elementos de ese subconjunto sea igual al parámetro **target**.

El parámetro **start** funciona como un contador y representa un índice en el arreglo de números **nums**. Ayúdenle a completarlo, por favor, si es tan amable.

Como un ejemplo, si pregunta si es hay un subconjunto de [2,4,8] que sume 10, la respuesta es verdadero. **Como otro ejemplo**, si preguntamos si hay un subconjunto de [2,4,8] que sume 20, la respuesta es falso.

```
private static boolean sumaGrupo(int start, int[] nums, int
target) {

}

public static boolean sumaGrupo(int[] nums, int target) {
    return sumaGrupo(0, nums, target);
}
```

3. Escriban un programa que calcule las combinaciones de letras de una cadena. Una combinación es un subconjunto de los n elementos, independiente del orden. Existen 2^n subconjuntos. Como un ejemplo, para "abc", debe dar esta respuesta:

a ab abc ac b bc c

```
public static void combinations(String s) {
    combinationsAux("", s);
}
```

Ayudas para resolver los Ejercicios

Ayudas para el Ejercicio 1..... [Pág. 4](#)

Ayudas para el Ejercicio 2..... [Pág. 5](#)

Ayudas para el Ejercicio 3..... [Pág. 6](#)

Ayudas para el Ejercicio 1



Pista 1: Condición de parada.



Pista 2:

Paso 1:

```
public static int gcd(int p, int q) {  
  
}
```



Pista 3: En el documental de Algoritmos de la BBC, en <https://www.youtube.com/watch?v=Q9HjeFD62Uk>, en el minuto 6:19, se muestra gráficamente el funcionamiento de este algoritmo.:



Pista 4: Este simulador puede ayudarles a entender <https://visualgo.net/recursion>



Error Común:

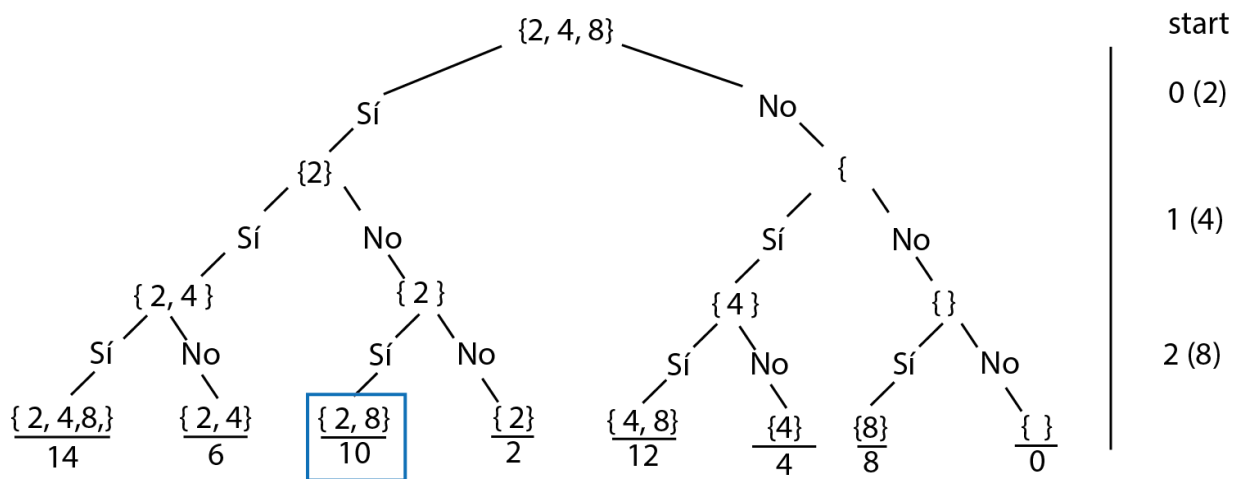


DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

Ayudas para el Ejercicio 2



Pista 0: Se deben hacer dos llamados recursivos. **Como un ejemplo,** a continuación, se muestra el árbol de ejecución para encontrar si un subconjunto de {2,4,8} suma 10.



Error Común 1: Este algoritmo falla porque al llamarse recursivamente con el parámetro `start` se queda en una recursión infinita.

```

public boolean sumaGrupo(int start, int[] nums, int target) {
    if (start >= nums.length) return target == 0;
    return sumaGrupo(start+1, nums, target - nums[start])
        || sumaGrupo(start, nums, target );
}
  
```



Error Común 2: Este algoritmo falla porque al llamarse recursivamente con el parámetro `start-1` se sale del arreglo cuando `start = 0`

```
public boolean sumaGrupo(int start, int[] nums, int target) {  
    if (start >= nums.length) return target == 0;  
    return sumaGrupo(start+1, nums, target - nums[start])  
        || sumaGrupo(start-1, nums, target );  
}
```



Error Común 3: Este algoritmo falla porque al llamarse recursivamente con el parámetro `start` se sale del arreglo cuando `start = 0`.

```
public boolean sumaGrupo(int start, int[] nums, int target) {  
    if (start >= nums.length) return target == 0;  
    return sumaGrupo(start+1, nums, target - nums[start])  
        || sumaGrupo(start+1, nums, target - nums[start - 1] );  
}
```

Ayudas para el Ejercicio 3



Pista 1: Calculen los subconjuntos, en una hoja de papel, para la cadena “abcd”



Pista 2: Si tienen dudas sobre cuáles son los subconjuntos de un conjunto, vean este video <https://www.youtube.com/watch?v=Bxv2Kvlltxs>. No se preocupen por el orden en que obtienen las combinaciones

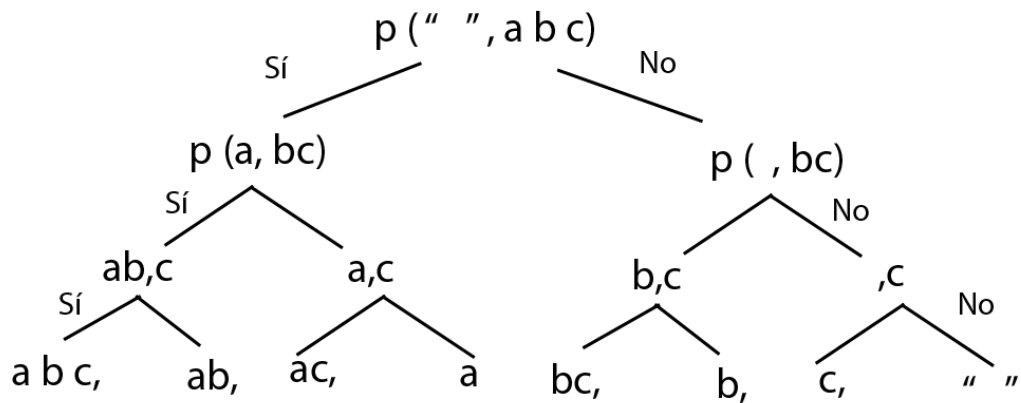


Pista 3: Primero defina una función auxiliar de esta forma:

```
private static void combinationsAux(String base, String s)  
{ ...  
}
```



Pista 4: La función recursiva debe generar el siguiente árbol de ejecución para generar los subconjuntos de la cadena “abc”.



Error Común: Un error común es calcular los prefijos en lugar de los subconjuntos, como se muestra en el siguiente ejemplo. La solución es hacer 2 llamados recursivos y no uno solo.

```

public static void prefijos(String base, String s){
    if (s.length() == 0){
        System.out.println(base);
    }
    else {
        prefijos (base + s.charAt(0), s.substring(1));
        System.out.println(base);
    }
}

```

¿Alguna inquietud?

CONTACTO

Docente Mauricio Toro Bermúdez

Teléfono: (+57) (4) 261 95 00 **Ext.** 9473

Correo: mtorobe@eafit.edu.co

Oficina: 19- 627

Agende una cita con él a través de **<http://bit.ly/2gzVg10>** , en la pestaña *Semana*. Si no da clic en esta pestaña, parecerá que toda la agenda estará ocupada.